Toward an optimal PRNN-based nonlinear predictor

Toward an Optimal PRNN-Based Nonlinear Predictor

Danilo P. Mandic and Jonathon A. Chambers, *Senior Member, IEEE*

***Abstract*— We present an approach for selecting optimal parameters for the pipelined recurrent neural network (PRNN) in the paradigm of nonlinear and nonstationary signal prediction. Although there has recently been progress in terms of algorithms for training the PRNN, no account has been made of some inherent features of the PRNN. We therefore provide a study of the role of nesting, which is inherent to the PRNN architecture. The corresponding number of nested modules needed for a certain prediction task, and their contribution toward the final prediction gain (PG) give a thorough insight into the way the PRNN performs, and offers solutions for optimization of its parameters. In particular, nesting, which is a contractive function by its nature, allows the forgetting factor in the cost function of the PRNN to exceed unity, hence it becoms an emphasis factor. This compensates for the small contribution of the distant modules to the prediction process, due to nesting, and helps to circumvent the problem of vanishing gradient, experienced in RNN's for prediction. The PRNN, with its parameters chosen based upon the established criteria, is shown to outperform the linear least mean square (LMS) and recursive least squares (RLS) predictors, as well as previously proposed PRNN schemes, at no expense of additional computational complexity.**

***Index Terms*—Forgetting factor, nesting, nonlinear prediction, PRNN, RNN.**

I. Introduction

A CLASS of physical signals, such as speech, is generated from a nonlinear mechanism, and has statistically nonstationary properties, which makes the task of their prediction difficult. Linear adaptive structures for prediction, such as least mean square (LMS) and recursive least squares (RLS) predictors do not account for the inherent nonlinearity in such signals, and as such face difficulties in providing reliable prediction. A nonlinear structure suitable for nonparametric prediction of nonlinear and nonstationary signals is the artificial neural network (ANN). In 1995, Haykin and Li [1] presented a novel, computationally efficient nonlinear predictor based on the pipelined recurrent neural network (PRNN). The PRNN consists of a number of small scale recurrent neural networks (RNN's), but maintains its relatively low computational complexity considering the entire number of neurons in its architecture. In addition, the PRNN architecture helps to circumvent the problem of vanishing gradient [2], due to: 1) creating a spatial representation of a temporal pattern, 2) putting time delays into the neurons or their connections, and 3) employing recurrent connections [3], [4]. In other words, the modular structure employed by the PRNN enables memory to be embedded into the PRNN [5]–[7], as well as the representation of block cascaded systems, such as the Wiener–Hammerstein system [4], [8], [9].

The learning algorithm which was used by Haykin and Li for the PRNN was a gradient descent (GD)-based algorithm known as the real-time recurrent learning algorithm (RTRL) [10], [11]. However, the variant of this algorithm used for training the PRNN suffers from some serious drawbacks. In [12] and [13], an improved RTRL-based algorithm was presented, together with the extended Kalman filter (EKF)-based algorithm for training the PRNN. Both of them outperformed the originally proposed algorithm.

As adaptation of parameters of the PRNN is a complex and demanding nonlinear optimization task, there is a need to have further insight into some inherent features of the PRNN which could yield even better performances using well-known strategies. One way would be to find some relationships between the parameters employed in learning [14], which is rather difficult for the PRNN. Hence, we analytically describe the core of the features of the PRNN for the prediction application, such as the nature and value of the forgetting factor, and the role of the number of modules in the PRNN, and offer a solution to obtain the best possible predictor in that environment.

This paper is organized in the following manner. In Section II, the PRNN-based nonlinear predictor is described, starting from the RNN, and concluding with the Haykin–Li's nonlinear predictor [1]. In Section III, the effect of nesting, which is inherent to the PRNN, is shown, and the influence of nesting on the output of the PRNN is elaborated. Furthermore, in Section IV, the role of the influence of the forgetting factor to the gradient-based learning of the PRNN is presented, and a new solution for weighting of the modules in the PRNN is proposed. In Section V, the performance of the proposed scheme is compared to the performances of known schemes [1], [12], [13], and it is shown that the proposed scheme substantially outperforms the existing ones. Finally, in Section VI, the main results presented in the paper are summarized.

II. The Pipelined Recurrent Neural Network (PRNN)

The PRNN is a modular neural network, and consists of a certain number M of RNN's as its modules, with each module consisting of N neurons. The structure of a single RNN is shown in Fig. 1. The RNN consists of three layers:

- input layer
- processing layer

Manuscript received January 15, 1998; revised January 26, 1999 and June 29, 1999.

D. P. Mandic is with the School of Information Systems, University of East Anglia, Norwich NR4 7TJ U.K.

J. A. Chambers is with the Communications and Signal Processing Research Group, Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, London, U.K.

Publisher Item Identifier S 1045-9227(99)09113-4.

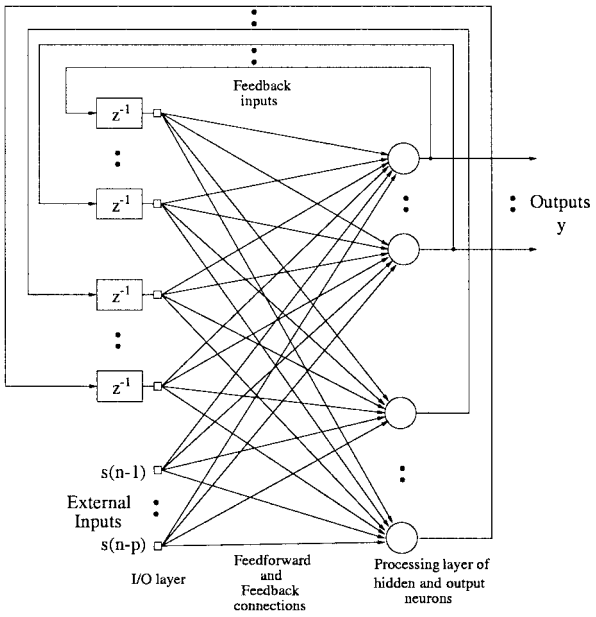

Fig. 1. Single recurrent neural network.

- output layer.

For each neuron k, $k \in \{1, N\}$, the elements u_i, $i \in \{1, p+F+1\}$, of the input vector to a neuron $\mathbf{u}$ (4), are weighted, then summed to produce an internal activation function of a neuron v (3), which is finally fed through a nonlinear activation function Φ (1), to form the output of the kth neuron y_k (2). The function Φ itself, is typically a monotonically increasing sigmoid logistic function, whose amplitude lies in the interval $(0, 1)$, and is given by

$$\Phi(v) = \frac{1}{1+\exp(-\beta v)}. \tag{1}$$

For the kth neuron, its weights form a $(p+F+1)\times 1$ dimensional weight vector $\mathbf{w}_k^T = [w_{k,1}, \cdots, w_{k,p+F+1}]$, where p is the number of external inputs and F is the number of feedback connections, one remaining element of the weight vector $\mathbf{w}$ being for the bias input weight. The feedback connections represent the delayed output signals of the RNN. In the case of the network shown in Fig. 1, we have $N = F$. Such a network is called a fully connected recurrent neural network (FCRNN). For more details about recurrent neural networks, refer to the landmark paper by Williams and Zipser [10]. The following equations fully describe the FCRNN:

$$y_k(n) = \Phi(v_k(n)), \qquad k \in [1, N] \tag{2}$$

$$v_k(n) = \sum_{l=1}^{p+N+1} w_{k,l}(n) u_l(n) \tag{3}$$

$$\mathbf{u}_i^T(n) = [s(n-1), \cdots, s(n-p), 1, y_1(n-1), y_2(n-1), \cdots, y_N(n-1)] \tag{4}$$

where the $(p+N+1)\times 1$ dimensional vector $\mathbf{u}$ comprises both the external and feedback inputs to a neuron, with vector $\mathbf{u}$ having "unity" for the constant bias input. Although the general network shown in Fig. 1 contains hidden neurons, whose outputs are not visible in the network output, but fed back to form the input vector, in the further analysis, only the case of all the neurons being visible will be considered.

In the PRNN configuration, the M modules, which are FCRNN's, are connected as shown in Fig. 2. The uppermost module of the PRNN, denoted by M, is simply an FCRNN, whereas in modules $\{M-1, \cdots, 1\}$, the only difference is that the feedback signal of the uppermost neuron output within the module m, denoted by $y_{m,1}$, $m = 1, \cdots, M-1$, is replaced with the appropriate output signal $y_{m+1,1}$, $m = 2, \cdots, M$ from its left neighbor module $m+1$. The $(p\times 1)$ dimensional external signal vector $\mathbf{s}^T(n) = [s(n-1), \cdots, s(n-p)]$ is delayed by m time steps ($z^{-m}\mathbf{I}$) before feeding the module m, where z^{-m}, $m = 1, \cdots, M$ denotes the m-step time delay operator, and $\mathbf{I}$ is the $(p\times p)$ dimensional identity matrix. The weight vectors $\mathbf{w}_k$ of each neuron k, are embodied in an $(p+N+1)\times N$ dimensional weight matrix $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_N]$, with N being the number of neurons in each module. All the modules operate using the same weight matrix $\mathbf{W}$. The overall output signal of the PRNN is $y_{\text{out}}(n) = y_{1,1}(n)$, i.e., the output of the first neuron of the first module. A full mathematical description of the PRNN is given in the following equations:

$$y_{i,k}(n) = \Phi(v_{i,k}(n)) \tag{5}$$

$$v_{i,k}(n) = \sum_{l=1}^{p+N+1} w_{k,l}(n) u_{i,l}(n) \tag{6}$$

$$\mathbf{u}_i^T(n) = [s(n-i), \cdots, s(n-i-p+1), 1, y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)] \quad \text{for} \quad 1 \le i \le M-1 \tag{7}$$

$$\mathbf{u}_M^T(n) = [s(n-M), \cdots, s(n-M-p+1), 1, y_{M,1}(n-1), y_{M,2}(n-1), \cdots, y_{M,N}(n-1)] \quad \text{for} \quad i = M. \tag{8}$$

Given the input vectors $\mathbf{u}_i(n)$ for each module i, $i \in [1, M]$ at the time instant n, the outputs of all the neurons in the network can be calculated using the equations given above.

At the time step n, for each module i, $i = 1, \cdots, M$, the one-step forward prediction error $e_i(n)$ associated with a module, is then defined as a difference between the desired response of that module $s(n-i+1)$, which is actually the next incoming sample of the input speech signal, and the actual output of the ith module $y_{i,1}(n)$, of the PRNN, i.e.,

$$e_i(n) = s(n-i+1) - y_{i,1}(n). \tag{9}$$

Since the PRNN consists of M modules, a total of M forward prediction error signals are calculated. The goal is to minimize some measure of the error in the entire PRNN, termed a *cost function*, which was originally proposed as a weighted sum of all the error signals from individual modules [1]. In such a performance criterion, a *forgetting factor* λ, $\lambda \in (0, 1]$, is introduced which determines the weighting of the individual modules. Thus, the overall cost function of the PRNN becomes

$$E(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{10}$$

where $e_i(n)$ is defined in (9).

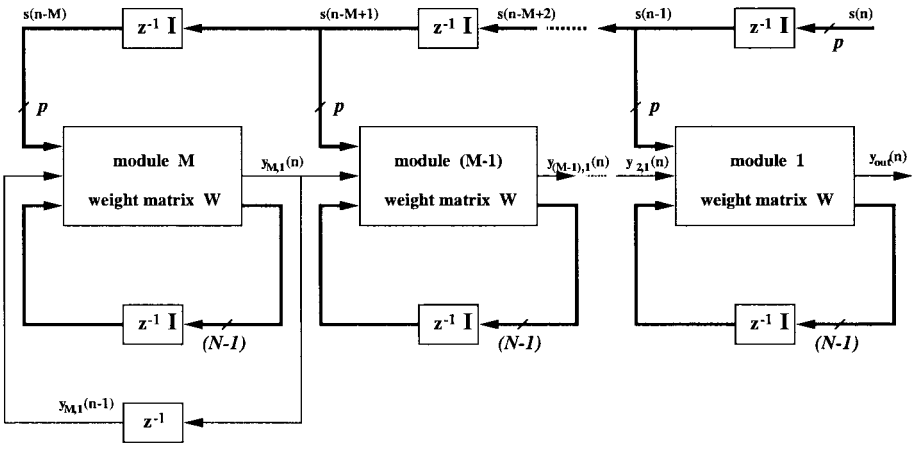

Fig. 2. Pipelined recurrent neural network.

Since the predictor operates on the nonstationary input data, a learning algorithm has to be chosen which, at each time step, calculates the weight correction factor $\Delta\mathbf{W}$ in order to update the weight matrix $\mathbf{W}$. Hence, the updated weight matrix at time-step $(n+1)$ can be calculated as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \Delta\mathbf{W}(n). \tag{11}$$

The merit of a pipelined recurrent neural network as compared to a single fully connected recurrent neural network is that its computational complexity is considerably reduced for the same total number of neurons. Let an FCRNN contain a total of N neurons; if M FCRNN's constitute modules of the PRNN, then the total number of neurons in the PRNN is $M \times N$. Having in mind that the computational complexity of an FCRNN trained with the gradient descent algorithm increases with $\mathcal{O}(N^4)$ [10], then the PRNN approach reduces the computational complexity of an entire network containing $M \times N$ neurons to a mere $\mathcal{O}(M \times N^4)$ [1]. Another advantage of the PRNN over an FCRNN is its increased capability of tracking time varying nonlinearity, and therefore the associated higher order statistics (HOS) of the probability density function (pdf) of the underlying process, owing to the connection of M modules containing FCRNN's as their architectural components.

A. The Haykin–Li's Nonlinear Predictor

The original approach was to combine the PRNN as a nonlinear part of an entire nonlinear predictor, which feeds the LMS linear predictor to obtain the predicted data. That procedure was composed of the three following subtasks.

- *Prediction:* Compute the one-step forward nonlinear prediction errors of the PRNN at the time instant n, using the procedure described above and (9).
- *Weight Updating:* A learning algorithm uses the suitably chosen overall cost function $E(n)$ (10) in order to calculate the weight matrix correction factor $\Delta\mathbf{W}$ which updates the weight matrix $\mathbf{W}$, as shown in (11).
- *Filtering:* Using (5)–(8) the output of the PRNN is computed. The updated input signal $\mathbf{u}_i(n+1)$ to every module i, $1 \le i \le M$ is formed by substituting the external signal input (speech) $\mathbf{s}_i(n) = [s(n-i), \cdots, s(n-i-p+1)]$ with the updated external signal input $\mathbf{s}_i(n+1) = [s(n-i+1), \cdots, s(n-i-p+2)]$.

The output of the PRNN was then fed into the LMS filter in order to produce the predicted signal of the nonlinear predictor. As our aim is to improve the performance of the PRNN part, and the LMS linear predictor was shown to contribute with approximately 2 dB toward the total prediction gain [12], [13], we shall concentrate on the PRNN part of the nonlinear predictor only.

III. The Effects of Nesting

The PRNN architecture provides nesting of the nonlinearities comprised in the modules of the PRNN. That being the case, the functional dependence of the output of the network $y_{1,1}(n)$ can be expressed as

$$\begin{aligned}\hat{s}_1(n) &= y_{1,1}(n)\\ &= \Phi(\mathbf{s}(n-1),\, y_{2,1}(n))\\ &= \Phi(\mathbf{s}(n-1),\, \Phi(\mathbf{s}(n-2),\, y_{3,1}(n))\\ &= \cdots\\ &= \Phi(\mathbf{s}(n-1),\, \Phi(\mathbf{s}(n-2),\, \cdots,\\ &\qquad \Phi(\mathbf{s}(n-M),\, y_{M,1}(n-M))\cdots)\end{aligned} \tag{12}$$

where it was assumed that all the neurons in the network operate with the same activation function Φ, and for the sake of simplicity, the functional dependence of the weight matrix $\mathbf{W}$ to the nested nonlinearities was omitted. The result given in (12) gives the PRNN its enhanced computing power compared to the conventional RNN. Indeed, it is a universal approximator in the sense that a PRNN with appropriate training can approximate any nonlinear autoregressive moving average (NARMA) process to any desired degree of accuracy, provided that a sufficient number of hidden neurons is available [15], [16]. The nested nonlinearity principle for an example of the logistic nonlinearity is shown in Fig. 3. As shown in Fig. 3, due to contractivity of the nonlinear activation function and modularity of the PRNN [17], the nesting process introduces a deteriorating effect in the relative amplitude of the output $y_{i,1}$, $i = 2, \cdots, M$ of a distant module i, when progressed through the PRNN. Thus, the relative contribution of the

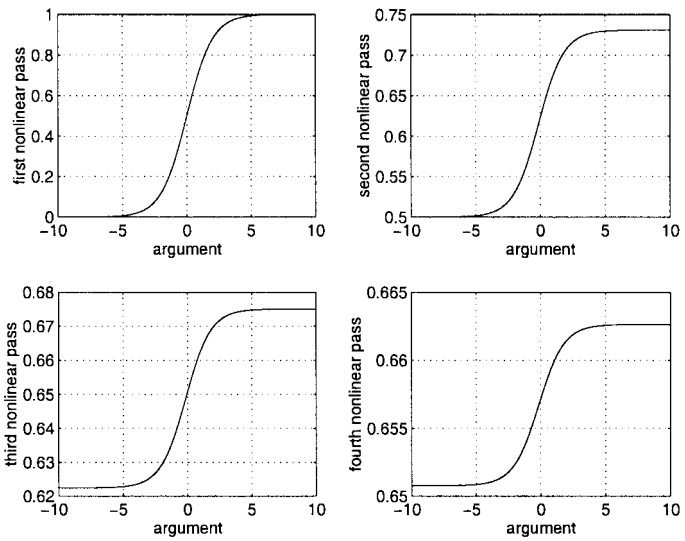

Fig. 3. Nested *logistic* nonlinearity.

output of the fourth module to the total amplitude at the output of the PRNN, when that information reaches the first module of the PRNN, has an amplitude $\in (0.65, 0.665)$, which has only a small impact on the final value of the predicted sample at the output of the PRNN. On the other hand, *nesting* does not have any influence on the values of the gradient of the activation function Φ, when progressed through the PRNN, which means that the dynamics of the learning process are not in any way affected by nesting, as shown in Fig. 3.

Now, let us provide insight into the values of the gradient for the nonlinear logistic activation function (1). The first derivative of the logistic function, which represents a gradient of the activation function in terms of the learning algorithms based upon (1) is given by

$$\Phi'(x) = \frac{\beta e^{-\beta x}}{(1+e^{-\beta x})^2}. \tag{13}$$

The maximum value of the function (13) for the common choice of $\beta = 1$ is 0.25. As the nested structure from (12) actually comprises the weight matrix $\mathbf{W}$, the output of the PRNN, which is in effect a conditional mean predictor, can be expressed as

$$\begin{aligned}\hat{s}_1(n) &= y_{1,1}(n) = \Phi(\mathbf{s}(n-1), \mathbf{W}(n), y_{2,1}(n)) \\ &= \Phi(\mathbf{s}(n-1), \mathbf{W}(n), \Phi(\mathbf{s}(n-2), \mathbf{W}(n), y_{3,1}(n)) \\ &= \Phi(\mathbf{s}(n-1), \mathbf{W}(n), \Phi(\mathbf{s}(n-2), \cdots, \\ &\quad \Phi(\mathbf{s}(n-M), \mathbf{W}(n), y_{M,1}(n-M))\cdots). \end{aligned} \tag{14}$$

In order to measure the influence of the output of a distant module $y_{i,1}(n)$, $i = 2, \cdots, M$ of the PRNN to its overall output $y_{1,1}(n)$ (Fig. 2), let us observe the derivative of the $y_{1,1}$ with respect to $y_{i,1}$, $i = 2, \cdots, M$, i.e., $\partial y_{1,1}/\partial y_{i,1}$. That is a measure of sensitivity of the output of the PRNN to the output of its ith module, denoted by $S_{y_{i,1}}^{y_{1,1}}$.

$$\begin{aligned} S_{y_{i,1}}^{y_{1,1}} &= \frac{\partial \Phi(\mathbf{s}_1, \mathbf{W}, \Phi(\mathbf{s}_2, \cdots, \Phi(\mathbf{s}_i, y_{i,1})\cdots)}{\partial y_{i,1}} \\ &= \dot{\Phi}(v_1)\frac{\partial v_1}{\partial y_{2,1}}\dot{\Phi}(v_2)\frac{\partial v_2}{\partial y_{3,1}}\cdots\dot{\Phi}(v_i)\frac{\partial v_{i-1}}{\partial y_{i,1}} \\ &= \dot{\Phi}(v_1)w_{1,1}\dot{\Phi}(v_2)w_{1,1}\cdots\dot{\Phi}(v_{i-1})w_{1,1}. \end{aligned} \tag{15}$$

Notice, that the maximum value for $\dot{\Phi}(v_i)$, $i = 1, \cdots, M$ is 0.25 as shown in (13). Hence, the upper bound of an estimate of $S_{y_{i,1}}^{y_{1,1}}$ becomes

$$\max_{\dot{\Phi}_{\mathrm{MIN}} \le \dot{\Phi} \le \dot{\Phi}_{\mathrm{MAX}}} (S_{y_{i,1}}^{y_{1,1}}) = \dot{\Phi}_{\mathrm{MAX}}^{i-1} w_{1,1}^{i-1} = 0.25^{i-1} w_{1,1}^{i-1}. \tag{16}$$

In our experiments, the values of $w_{1,1}$ were such that $|w_{1,1}| \ll 1$, which gives

$$\sup(S_{y_{i,1}}^{y_{1,1}}) = 0.25^{i-1}. \tag{17}$$

For $M = 5$, $\sup(S_{y_{M,1}}^{y_{1,1}}) = 0.25^4 = 0.0039$ which is the upper bound of the influence of the amplitude of $y_{M,1}$ to the amplitude of $y_{1,1}$. That is a numerical measure of the nesting effect, shown in Fig. 3. That is also the explanation why Haykin and Li [1] could not achieve remarkably improved prediction gains while increasing the number of modules for $M > 5$. It is to be noticed that the above result (17) does not depend on the forgetting factor λ. Since speech is a heavily correlated signal, although nonstationary, and the computational complexity of the PRNN grows with the increase in the number of its modules, the $p+M-1$ external input speech samples in the PRNN are in most feasible cases such that $p+M-1 < 10$, otherwise the network grows too complex, since its computational complexity increases with $\mathcal{O}(M \times N^4)$. We can assume that the $p+M+1$ speech samples belong to a section of speech over which piecewise stationarity can be assumed. Moreover, there is an information flow between modules of the PRNN, where less distant modules accept information of a predicted value of speech from their neighbor module. Therefore, it seems reasonable to increase the influence of the distant modules to the learning process, more than allowed by the cost function (10), especially with M low. Hence, there is a need to find another way to make the distant modules play their full role in the nonlinear predictor. There are two ways of how a distant module can make an influence to the overall output of the PRNN.

1) Through its output amplitude, due to *nesting.*
2) Through the learning process, since the overall correction $\Delta\mathbf{W}$ to the common weight matrix $\mathbf{W}$ is calculated over all modules, according to the *cost function* of the network.

Since nothing can be done to improve the influence of a distant module amplitude to the output of the PRNN through *nesting*, it appears that the only way of improving the influence of distant modules to the prediction process is through the *cost function* of the PRNN (10), i.e., through the process of learning.

IV. An Analysis of the Influence of the Forgetting Factor to the Total Prediction Gain

The cost function for the PRNN has traditionally been defined as

$$E(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n). \tag{18}$$

Hence, the elements of the correction $\Delta\mathbf{W}(n)$ to the weight matrix update $\mathbf{W}(n+1)$ can be calculated as

$$\begin{aligned}\Delta w_{k,l}(n) &= -\eta\frac{\partial}{\partial w_{k,l}(n)}\left(\sum_{i=1}^{M}\lambda^{i-1}e_i^2(n)\right)\\ &= -2\eta\sum_{i=1}^{M}\lambda^{i-1}e_i(n)\frac{\partial e_i(n)}{\partial w_{k,l}(n)}.\end{aligned}\tag{19}$$

Having in mind the nature of the elements of the sum (19), it can be seen as a weighted sum of the correction factors due to individual modules, namely

$$\begin{aligned}\Delta w_{k,l}(n) &= -\eta\frac{\partial}{\partial w_{k,l}(n)}\left(\sum_{i=1}^{M}\lambda^{i-1}e_i^2(n)\right)\\ &= -2\eta\sum_{i=1}^{M}\lambda^{i-1}\Delta w_{k,l}^{i}(n)\end{aligned}\tag{20}$$

or, equivalently

$$\Delta\mathbf{W}(n)=\sum_{i=1}^{M}\lambda^{i-1}\Delta\mathbf{W}_i(n)\tag{21}$$

where $\Delta w_{k,l}^{i}$ represents the correction factor of a single weight $w_{k,l}$ due to each individual module i, $1\le i\le M$, and $\Delta\mathbf{W}_i$ represents the corresponding weight matrix correction due to module i. Therefore, the total correction to the weight matrix $\mathbf{W}$, can be bigger in magnitude than any of the individual corrections making the sum (20). We have already seen that the nesting process (12) affects the contribution of the relative amplitude of the output of a distant module to the overall output of the network $y_{1,1}$, but *does not* affect the learning process. Moreover, it is the forgetting factor λ that has an influence on the learning process. As seen from (21), although every module of the PRNN has to calculate its full contribution $\Delta\mathbf{W}_i$ to the overall correction of the weight matrix $\Delta\mathbf{W}$, it becomes further scaled by multiplying with λ^{i-1}. For the Mth module, e.g., $M=5$, its contribution to the correction of the overall weight matrix is multiplied by λ^4, which for $\lambda=0.9$, as in the Haykin–Li's paper [1] equals 0.6561. Hence, not only the amplitude of a distant module does not have significant influence on the amplitude of the output of the PRNN, but also the forgetting factor in the cost function lowers the contribution of the correction to the weight matrix of a distant module, which discards its significance in two ways. However, the Mth module is the only one which is a proper fully connected RNN and does not involve any approximation in its structure. It emerges therefore, that the contribution $\Delta\mathbf{W}_i$, $i=2,\cdots,M$ to the total correction matrix $\Delta\mathbf{W}$ from distant modules should be somehow more evenly taken into account, when training the PRNN. One intuitive approach would be to amplify the contribution of distant modules to the learning process by raising the value of the forgetting factor λ, even for λ slightly bigger then unity, i.e., $\lambda>1$. In that case, we will refer to λ as an *emphasis factor*. Using the emphasis factor, the distant modules become heavily involved in the learning process of the PRNN. The use of an emphasis factor can be approved by the fact that, due to nesting, the influence of distant modules to the dynamics of the PRNN should indeed be amplified by some constant greater than unity.

On the other hand, the forgetting factor λ is RLS motivated, where the cost function is $E(n)=\sum_{\tau=0}^{n}\lambda^{\tau}e^2(n-\tau)$, with $0<\lambda\le 1$. In the case of the PRNN, however, the forgetting factor is introduced along the modules. Realizing that

$$\begin{aligned}y_{1,1}(n-1)&\neq y_{2,1}(n)\\ y_{2,1}(n-1)&\neq y_{3,1}(n)\\ &\cdots\\ y_{(M-1),1}(n-1)&\neq y_{M,1}(n),\end{aligned}\tag{22}$$

i.e., the outputs of the modules in the PRNN are not realizations of the same stochastic process, we can introduce an emphasis factor, which is a linear weighting factor, rather than a forgetting factor.

V. Experimental Results

Three different speech signals, denoted by $s1$, $s2$ and $s3$ were used to test the nonlinear predictor. Signal $s2$ was identical to that used in [1], whereas $s1$, $s2$, and $s3$ were used in [12]. The content of the speech signals used in simulations was as follows.

- $s1$: Speech sample "Oak is strong and ...," length 10 000, sampled at 8 kHz.
- $s2$: Speech sample "When recording audio data ...," length 10 000, sampled at 8 kHz.
- $s3$: Speech sample "I'll be trying to win ...," length 10 000, sampled at 11 kHz.

The signals have been made public and are available on the World Wide Web (WWW) from the author's homepage [18]. The amplitudes of the signals were adjusted to lie in the range of the function Φ, i.e., $\in(0,1)$. The measure that was used to assess the performance of the predictors was the forward prediction gain R_p given by

$$R_p\triangleq 10\log_{10}\left(\frac{\hat{\sigma}_s^2}{\hat{\sigma}_e^2}\right)\text{ dB}\tag{23}$$

where $\hat{\sigma}_s^2$ denotes the estimated variance of the speech signal $\{s(n)\}$, whereas $\hat{\sigma}_e^2$ denotes the estimated variance of the forward prediction error signal $\{e(n)\}$. This approach to the definition of prediction gain is different from the one used in [1], which used the mean squared values of the signal and error instead of appropriate variance estimates. The usage of variance estimates is preferable, though, because the dc term contained in the mean squared values leads to biased results.

A. The Initialization

The initialization of the weights $\mathbf{W}$ was achieved via epochwise training as is commonplace for neural networks with fixed weights. An initial weight matrix was chosen randomly. The first L samples of the input signal $\mathbf{s}$ were chosen as an input to the PRNN. The L samples were used for L weight update $\Delta\mathbf{W}$ calculations. Those L updates were summed to form an epoch weight update $\Delta\mathbf{W}_{\text{epoch}}$. Then,

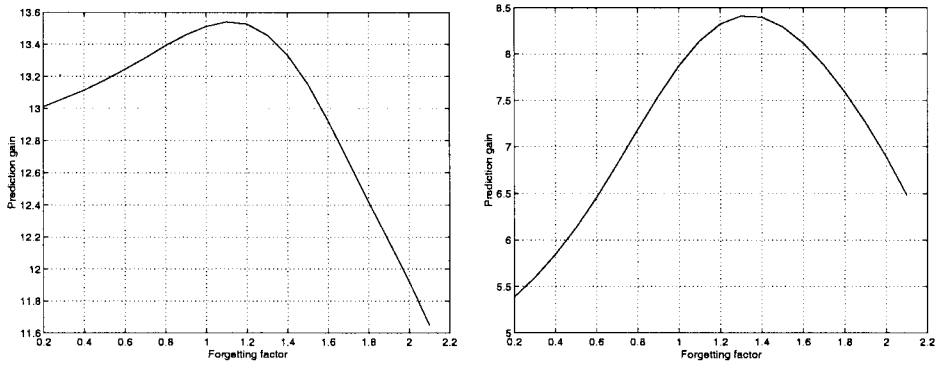

Fig. 4. Relationship between prediction gain Rp and forgetting factor λ for speech signals $s2$ and $s3$: (a) Prediction gain Rp versus the forgetting factor λ for $s2$ and (b) prediction gain Rp versus the forgetting factor λ for $s3$.

$\Delta\mathbf{W}_{\text{epoch}}$ was used instead of $\Delta\mathbf{W}$ to update $\mathbf{W}$. The whole procedure was then termed *an epoch*. L was chosen to be 300, and the number of epochs required was 200, the number of external speech inputs to a module was $p = 4$, and the number of neurons per module was $N = 2$ [1], [12], [13]. The slope of the activation function β was chosen to be unity, as in the previous related work, which preserves nesting, due to contractivity of the logistic activation function for $\beta < 4$ [17].

B. Experiments on Prediction of Speech

To confirm that the distant modules should be heavily involved in the learning process of the PRNN, an appropriate experiment was undertaken. A configuration used in [1], [12], and [13] was taken as a starting point, where the number of modules M in the PRNN considered as optimal was $M = 5$. Let us first consider the influence of the forgetting factor λ to the total prediction gain Rp. Our intuitive approach was that, due to the effects of nesting to the contribution of distant modules to the output of the PRNN, the values for λ, for a relatively small number of modules, can be even taken as $\lambda > 1$. The relationship between the prediction gain Rp and the value of the emphasis factor λ for speech signals $s2$, and $s3$, having the PRNN with $p = 4$, $M = 5$, $\eta = 0.07$, $N = 2$ is given in Fig. 4. From Fig. 4, the best value for the emphasis factor λ for the speech signal $s2$ is $\lambda_{opt} = 1.1$, where prediction gain $Rp = 13.54$ dB, and $\lambda_{opt} = 1.3$ for the speech signal $s3$ where $Rp = 8.41$ dB. The appropriate value of λ for the speech signal $s1$ was the same as for $s2$. The experiment totally approved our expectations that for a medium number of modules, such as typically $M = 5$, as in [1] and [13], the emphasis factor whose amplitude is slightly greater than unity should be used. Increasing the value of λ further, i.e., $\lambda > \lambda_{opt}$ leads to further deterioration in the value of prediction gain Rp. The learning rate η was chosen bigger than in [1], [12], and [13], which caused the prediction gain Rp to be a nonmonotonic function of the number of modules M. In Fig. 5, a relationship between prediction gain Rp and the number of modules M for the speech signals $s1$ and $s2$ is given. As expected, the prediction gain Rp showed a significant increase for up to a medium number of modules in the PRNN, whereas for a large number of modules, since $\lambda > 1$, it showed fast deterioration. Nevertheless, as using a large number of modules in the PRNN does not approve its usage, considering the dramatically increasing computational complexity of the PRNN with increasing the number of modules ($\mathcal{O}(M \times N^4)$), it is not likely that one will work with, e.g., $M > 5$. Moreover, the maximum prediction gain can be obtained for as small a number of modules as $M = 2$, in the case of the speech signal $s1$. Table I shows the comparison between the strategy with giving distant modules more significance by enlarging the factor λ, and Haykin–Li's experiment. In the first row of Table I the values of prediction gain Rp for three speech signals, when using only a sole linear LMS predictor is shown. The second row shows the prediction gain for the RLS linear predictor. Furthermore, the results obtained in [12] and [13] are shown in the third and fourth row, with SG + LMS denoting the use of the stochastic gradient algorithm in the PRNN, and the LMS algorithm afterwards, and ERLS + RLS denoting the use of the extended recursive least squares (ERLS) algorithm in the PRNN, and the RLS algorithm afterwards. The parameters used in the first four rows were $p = 4$, $N = 2$, $\lambda = 0.9$, $M = 5$. Notice that it was found that the linear predictors used after the PRNN in previous configurations, improved the prediction gain by about 2 dB [13]. In the sixth row, we show the prediction gains obtained at the output of the PRNN, for the case $M = 5$, and $\lambda = 1.3$ for the speech signals $s1$ and $s3$, and $\lambda = 1.1$ for the speech signal $s2$. As the prediction gain Rp in that case, according to Fig. 5, can achieve its maximum for $M < 5$, even for $M = 2$, the maximum values of prediction gains obtained for the speech signals are shown in the seventh row of Table I. For the signal $s2$ the maximum value of prediction gain was exactly $M = 5$. The results shown in Table I show that for $\lambda > 1$, as proposed, the corresponding predictors easily outperform the LMS predictor. Values for $s2$ and $s3$ even outperform the corresponding SG + LMS values. In the case where the maximal values for Rp were obtained (the last row in Table I), the PRNN predictor, as proposed, outperforms the LMS, RLS, and SG + LMS predictors [13], which are the stochastic gradient based predictors, whereas as compared to the ERLS + RLS predictor, it shows 2–2.5 dB worse performance for speech signals $s1$ and $s3$.

A cost function with an emphasis factor $\lambda > 1$, provides simultaneously both the error minimization and the penalty for complexity part, as desired in signal processing.

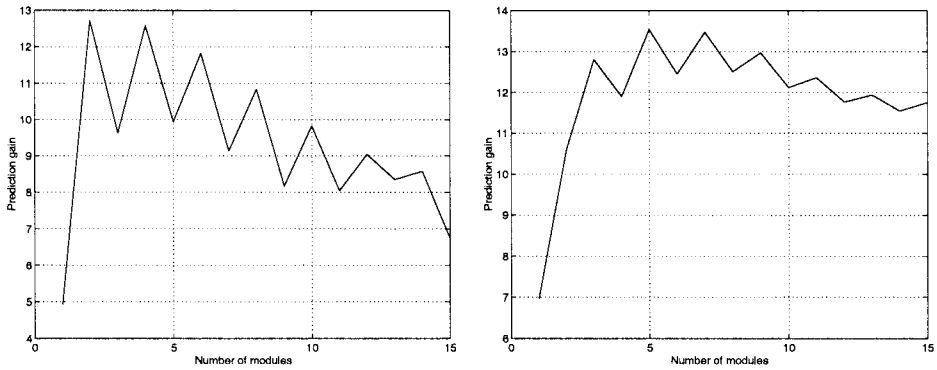

Fig. 5. Relationship between prediction gain Rp and number of modules M for $\lambda > 1$, and speech signals $s1$ and $s2$: (a) prediction gain Rp versus the number of modules M for $s1$ and (b) prediction gain Rp versus the number of modules M for $s2$.

TABLE I
COMPARISON BETWEEN PREDICTION GAINS Rp BETWEEN HAYKIN–LI'S SCHEME AND PROPOSED SCHEME

speech signal	$s1$	$s2$	$s3$
$Rp[dB]$ LMS only	9.24	8.06	6.31
$Rp[dB]$ RLS only	12.70	11.55	9.19
$Rp[dB]$ SG+LMS	10.25	9.49	7.30
$Rp[dB]$ ERLS + RLS	14.77	13.40	10.90
Modified PRNN only with $\lambda > 1$			
$Rp[dB]$ SG for $M = 5$	10.04	13.54	8.41
$Rp[dB]$ SG	12.72	13.54	8.53

VI. CONCLUSIONS

Insight into the core of the pipelined recurrent neural network (PRNN) in prediction applications is provided. Since modules of the PRNN perform simultaneously in a pipelined parallel manner, this leads to a significant improvement in the total computational efficiency of such a predictor. Modularity in the PRNN provides embedding, which helps to circumvent problems of vanishing gradient, experienced with RNN's. It is shown, that modules of the PRNN contribute to the final, predicted value at the output of the PRNN in two ways, namely through the process of nesting, and through the process of learning. A measure of the influence of the output of a distant module to the amplitude at the output of the PRNN was analytically found as the sensitivity $S^{y_1,1}_{y_i,1}$, $i = 2, \cdots, M$ and the upper bound for it was derived. That result was confirmed graphically for the example of the logistic nonlinearity. Furthermore, an analysis of the influence of the forgetting factor in the cost function of the PRNN to the process of learning was undertaken, where it was found that for the PRNN, the forgetting factor can even exceed unity in order to obtain the best predictor, becoming therefore an emphasis factor. The simulation on three speech signals supported that approach, and outperformed the other stochastic gradient-based schemes, and showed performance close to the extended Kalman filter-based schemes. Our intention was to consider only the nonlinear part, i.e., only the PRNN predictor, we showed that in that case, our approach outperforms the other stochastic gradient-based techniques for the PRNN, and even matches the results obtained by the more powerful extended recursive least squares (ERLS) algorithm.

REFERENCES

[1] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Processing*, vol. 43, pp. 526–535, 1995.
[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, pp. 157–166, 1994.
[3] S.-S. Kim, "Time-delay recurrent neural network for temporal correlations and prediction," *Neurocomputing*, vol. 20, pp. 253–263, 1998.
[4] D. P. Mandic and J. A. Chambers, "A nonlinear adaptive predictor realized via recurrent neural networks with annealing," in *Dig. Inst. Elect. Eng. Colloquium Statist. Signal Processing*, pp. 2/1–2/6, 1999.
[5] T. Lin, B. G. Horne, and C. L. Giles, "How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies," *Neural Networks*, vol. 11, pp. 861–868, 1998.
[6] T. Sauer, J. A. Yorke, and M. Casdagly, "Embedology," *J. Statist. Phys.*, vol. 65, pp. 579–616, 1991.
[7] C. Scheier, R. Pfeifer, and Y. Kunyioshi, "Embedded neural networks: Exploiting constraints," *Neural Networks*, vol. 11, pp. 1551–1569, 1998.
[8] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.
[9] A. D. Back and A. C. Tsoi, "Nonlinear system identification using multilayer perceptrons with locally recurrent synaptic structure," in *Proc. IEEE-SP Wkshp. NNSP II*, 1992, pp. 444–453.
[10] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.
[11] S. Haykin, *Neural Networks—A Comprehensive Foundation.* Englewood Cliffs, NJ: Prentice-Hall, 1994.
[12] D. P. Mandic, J. Baltersee, and J. A. Chambers, "Nonlinear prediction of speech with a pipelined recurrent neural network and advanced learning algorithms," in *Signal Analysis and Prediction*, A. Prochazka, J. Uhlir, P. J. W. Rayner, and N. G. Kingsbury, Eds. Boston: Birkhauser, 1998, pp. 291–309.
[13] J. Baltersee and J. A. Chambers, "Nonlinear adaptive prediction of speech signals using a pipelined recurrent neural network," *IEEE Trans. Signal Processing*, vol. 46, pp. 2207–2216, 1998.
[14] D. P. Mandic and J. A. Chambers, "Relationship between the slope of the activation function and the learning rate for the RNN," *Neural Comput.*, vol. 11, no. 5, pp. 1069–1077, 1999.
[15] L. K. Li, "Approximation theory and recurrent networks," in *Proc. Int. Joint Conf. Neural Networks*, 1992, vol. II, pp. 266–271.
[16] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, pp. 240–254, 1994.
[17] D. P. Mandic and J. A. Chambers, "Global asymptotic stability of nonlinear relaxation equations realized through a recurrent perceptron," in *Proc. Int. Conf. Acoust., Speech, Signal Processing (ICASSP-99)*, 1999, vol. 2, pp. 1037–1040.
[18] D. P. Mandic, personal homepage. Available http://www.dsp.ee.ic.ac.uk/~mandic

Danilo P. Mandic received the B.Sc. (Hons.) degree in automatic control and the M.Sc. degree in signal processing from University of Banja Luka, Bosnia-Herzegovina. He received the Ph.D. degree in nonlinear adaptive signal processing from Imperial College, London, U.K. His areas of interest are linear and nonlinear adaptive signal processing, neural networks, biomedical signal and image processing, system identification, stability theory, and computer vision.

He is currently a Lecturer in computer science at the School of Information Systems, University of East Anglia, Norwich, UK.

Dr. Mandic has received awards for his collaboration with industry and was also awarded a Nikola Tesla medal for his innovative work.

Jonathon A. Chambers (M'93) was born in Peterborough, U.K., in 1960. After an electronics artificer apprenticeship in the Royal Navy, he received the first class B.Sc. (Hons.) degree in electrical and electronic engineering from the Polytechnic of Central London, U.K., receiving the Robert Mitchell Medal as the top graduate in 1985. He received the Ph.D. degree in adaptive signal processing in 1990 after studying at Imperial College, London, U.K., and at Cambridge University, Cambridge, U.K.

He spent three years as a Research Scientist at Schlumberger Cambridge Research, applying adaptive signal processing techniques to oilfield-related applications. He returned to a lectureship in signal processing in the Department of Electrical and Electronic Engineering, Imperial College, in 1994 and was promoted to a readership in signal processing in 1998. He has authored and coauthored many technical publications on adaptive signal processing and its applications in mobile communication systems.

Dr. Chambers is a member of Institute of Electrical Engineers Professional Group Committee E5 on Signal Processing, a Guest Editor for the International Journal of Adaptive Control and Signal Processing, and has served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING.